NORTH SPINE CANTEEN SYSTEM APPLICATION

Project done by,
Aneez Ahmed Jaheezuddin
Jonathan Chan
Heon Jung Kim

INDEX

1. Aim

The aim of the North Spine Canteen System software is to design a system through which a customer can view store information, menu items, purchase details, and store timings for various eateries in the North Spine canteen.

2. Synopsis

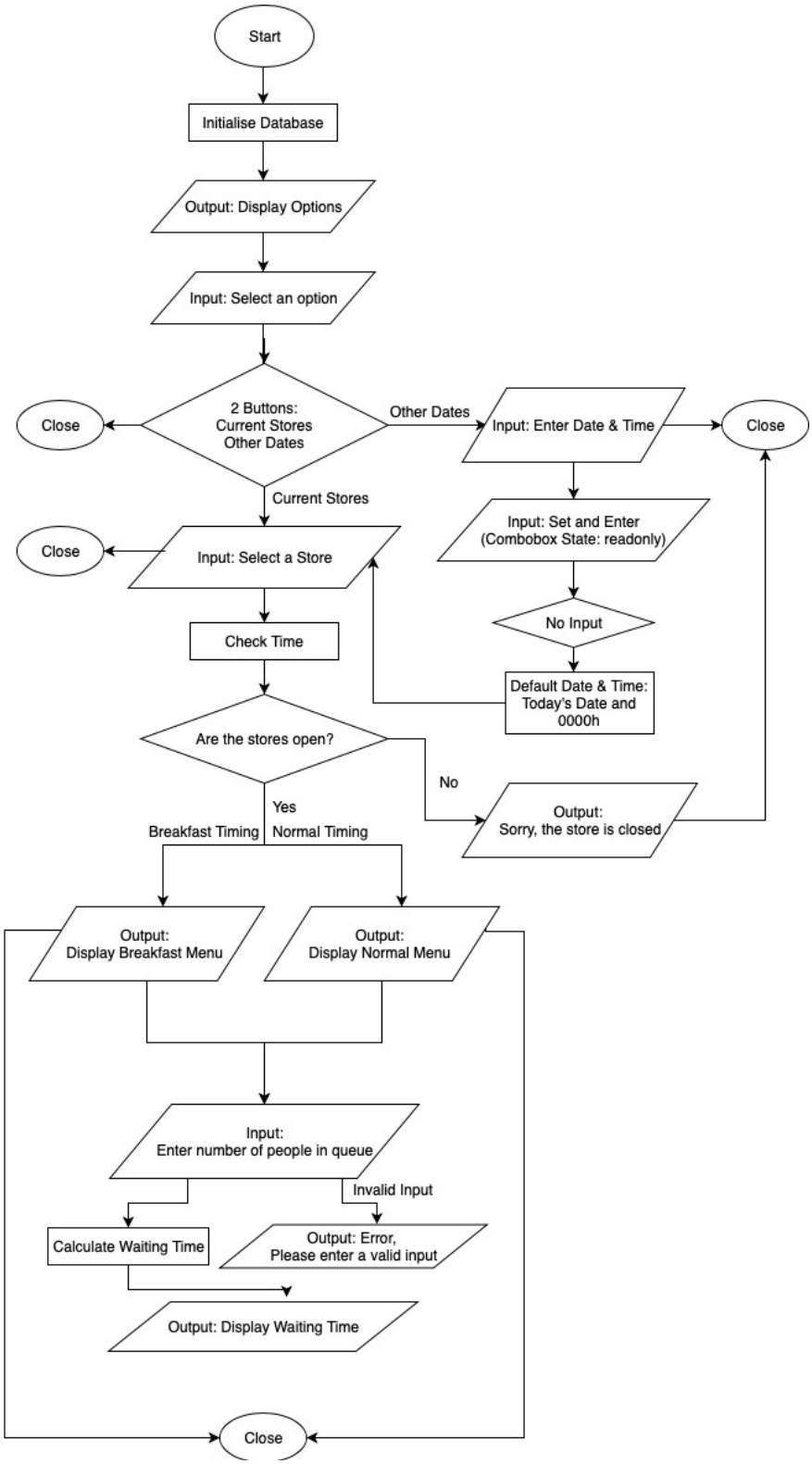
The objective of this project is to provide customers of the North Spine Canteen with convenient access to the canteen details and information using the proposed application. The application will contain a database that holds information regarding the canteen stores', menu items and pricing, and availability of the various items in each store. Hence, the application enables the automation of the existing manual enquiry system using computerized software in order to facilitate easier access, storage, and manipulation of canteen data.

Hence, the proposed application essentially provides a framework on how to manage the North Spine Canteen for better performance and services to its customers by allowing organization to make better utilization of its resources.

3. Uses of the Proposed System

The proposed application can implement an automated enquiry system for the North Spine Canteen. This will enable customers and owners to save more time and make better uses of available resources. All enquiries are maintained through the application database. Hence, the application will act as a mediator between the customers and the stores. Customers can access the required information from a single location.

4. System Flow Diagram



5. Header Files and User-Defined Data

5.1 Header Files (Fig. 1)

```
from tkinter import *
from tkinter import ttk
from PIL import ImageTk, Image
from tkmacosx import Button
try:
    import tkinter as tk
    from tkinter import ttk
except ImportError:
    import Tkinter as tk
    import ttk

from tkcalendar import Calendar, DateEntry
from tkinter.messagebox import showinfo
#the above import statements deal with creation and design of GUI using tkinter

import sys
import datetime #python library to retrieve and manipulate system and user defined date and time
```

Fig. 1 Header Files

For our program, we have utilized the tkinter module, PIL module for importing of images, tkmacosx for the visual enhancement for buttons, tkcalendar for the calendar widget and tkinter.messagebox to display pop-up windows.

5.2 User-Defined Data

The North Spine Canteen Program is built using the Tkinter module. The information of each store is stored within a .txt file (Fig. 2), with the item and price separated by a colon, and is displayed using the display_menu() function.

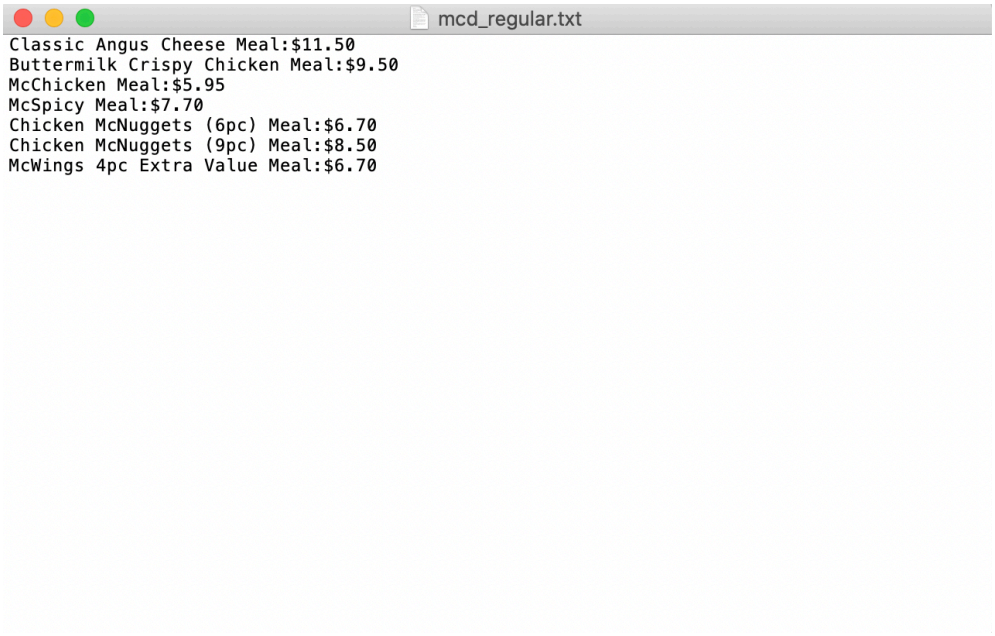


Fig.2 Sample .txt file for McDonald's Breakfast Menu

5.2.1 place_widgets()

place_widgets() allows us to place the labels and buttons at a fixed point for every store within the program (Fig. 3). This enables us to make the program more modular as lesser lines of codes are needed whereby there would not be any repetition within every individual store.

```
#function to define the placement of widgets at each storefront page
def place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg):
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    calcButton.place(x=590, y=328, height=36, width=70)
    waittime_result.place(x=530,y=375, height=20, width=40)
    qEntry.place(x=525,y=330, height=30, width=50)
    backButton.place(x=30, y=400, height=50, width=50)
    endProg.place(x=740, y=20, height =34, width=40)
```

Fig. 3 place_widgets()

5.2.2 display_menu()

This function allows the menu to be displayed within the program (Fig. 4). It invokes the initialization of the string which holds the text files' contents, which then converts the contents of the string into a dictionary with the menu item as the key and the price as the value of the dictionary. Then, final_str is defined to be the string that will be displayed on the menu label widget. Also, additional stylings such as the font is also taken into consideration to improve the visuals of the program (Fig. 5).

```
#function to display the menu items of a particular store
def display_menu(f, itemDict):
    line_str = f.readlines() #initialization of string which holds text file contents

    for i in line_str: #converts the contents of the string into a dictionary with the menu item as the key and the price as the value
        new_str = i.split(":")
        itemDict[new_str[0]] = new_str[1]

    final_str = "" #initialization of final string that will be used to display the menu label widget
    for key, value in itemDict.items():
        final_str += key + " : " + value

    itemLabel = ttk.Label(root, text = final_str, font=('Trattatello')) #menu items label initialization and styling
    itemLabel.place(x=280,y=180, width=250, height=125)
```

Fig. 4 display_menu()

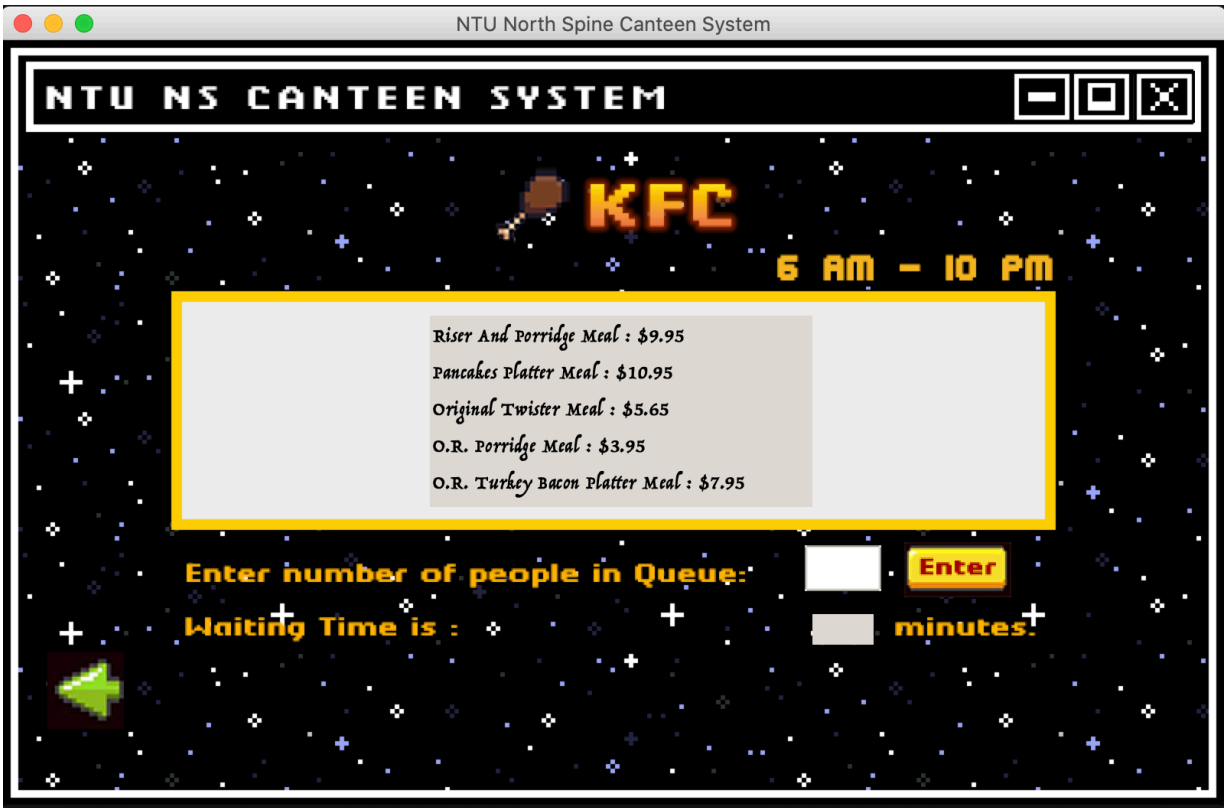


Fig 5. KFC's Breakfast Menu being displayed in the program

5.2.3 SetDateToCurr()

This function allows the program to retrieve the current system date and time (Fig 6 & 7).

```
#Function to set the program to the current system date and time
def SetDateToCurr():
    global day_str
    global time_str
    global hour_int
    now = datetime.datetime.now()
    day_str = datetime.datetime.today().strftime("%A")
    time_str = now.strftime("%H:%M:%S")
    hour_int = int(now.hour)
```

Fig. 6 SetDateToCurr()

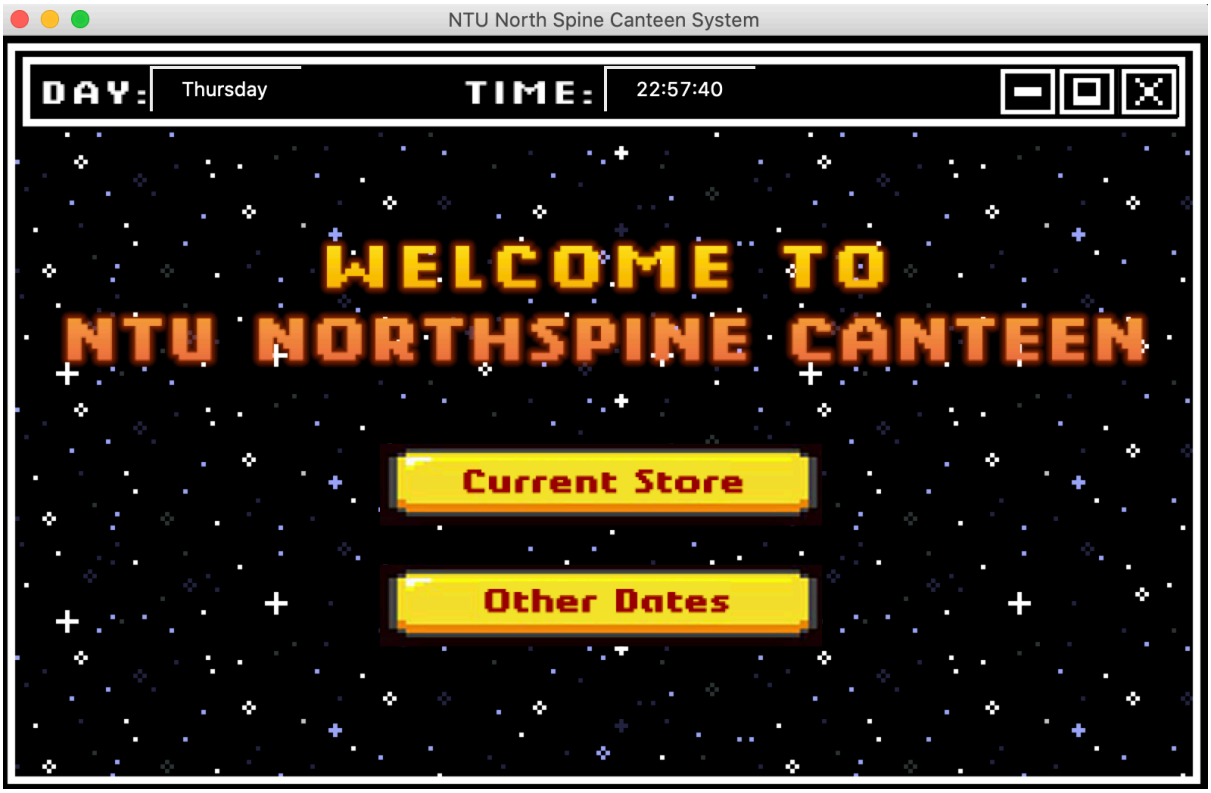


Fig. 7 Current date and time being displayed

5.2.4 SetDateToCustom(cus_day, cus_hour, cus_min)

This function allows the program to retrieve an input from the user for the custom date and time (Fig. 8). If the input is left empty, the current date and the default time of 0930h is set.

```
#Function to set the program to a user defined date and time
def SetDateToCustom(cus_day, cus_hour, cus_min):
    global day_str
    global time_str
    global hour_int

    #Special case handling to display the user defined time properly when it is a single digit
    if int(cus_hour) < 10:
        cus_hour = "0" + cus_hour

    if int(cus_min) < 10:
        cus_min = "0" + cus_min

    day_str = cus_day
    hour_int = cus_hour
    time_str = cus_hour + ":" + cus_min + ":00"
    hour_int = int(cus_hour)
```

Fig. 8 SetDateToCustom(cus_day, cus_hour, cus_min)

Also, if the user defines a timing with a single digit (e.g. 0905; Fig. 9), it is important to have the special case handling so as to display the time correctly as shown below (Fig. 10)

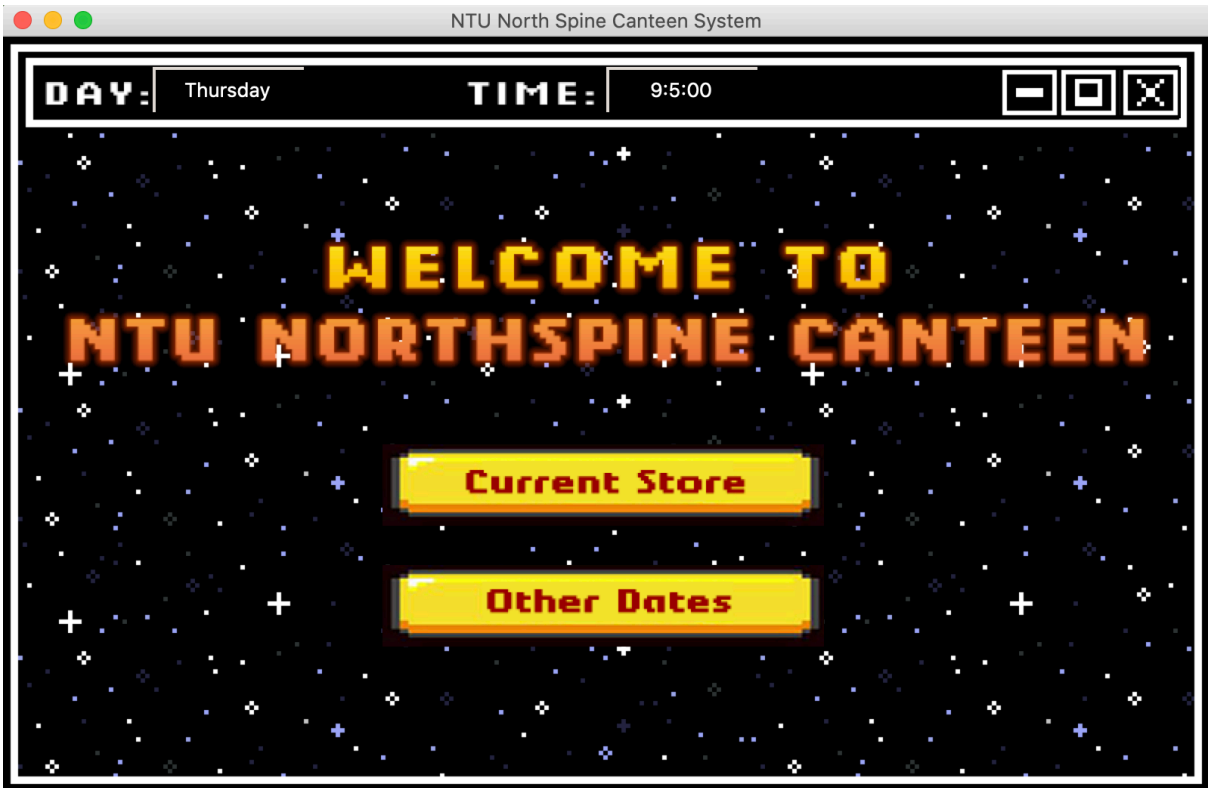


Fig. 9 Incorrect format of timing being shown without case handling

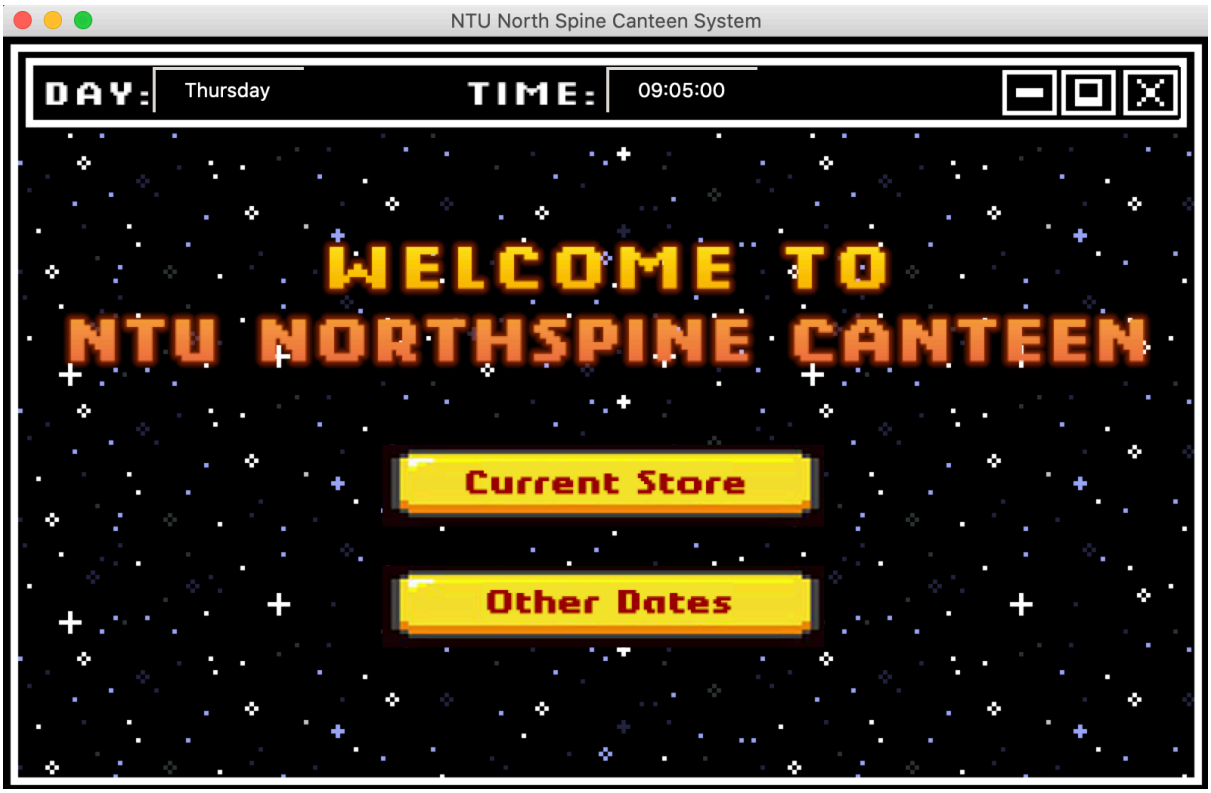


Fig. 10 Time correctly displayed with case handling

5.2.5 leavePage()

This function allows the function to destroy the root window and then exit the program (Fig. 11).

```
#Function to end program when user clicks exit
def leavePage():
    try:
        root.destroy()
        exit()
    except:
        pass
```

Fig. 11 leavePage()

5.2.6 dateEntryPage(*args) and getDateValues()

getDateValues() is a nested function within its enclosing function, dateEntryPage(*args). dateEntryPage defines a page which allows users to input custom date and time preferences, and hourString and minString is assigned to a string object to enable us to use the .get() function within the tkcalendar module which was imported earlier (Fig. 12)

getDateValues allows the system to retrieve the custom date and time values from the user, and the mindate and maxdate is set to define a date parameter window for the calendar widget.

Additional styling is also done to improve the visuals of the program (Fig. 13 & 14).

```
#Defines a page that allows users to input custom date and time preferences
def dateEntryPage(*args):
    hourString = StringVar()
    minString = StringVar()

    #Function to retrive the custom date and time values from the user
    def getDateValues():
        try:
            day = cal.selection_get()
            hour = hourString.get()
            minute = minString.get()
            SetDateToCustom(day.strftime("%A"), hour, minute)
        except ValueError:
            pass

    global flag
    flag = True

    today = datetime.date.today()
    mindate = datetime.date(today.year, today.month, today.day)#sets minimum date on the calendar as todays date
    maxdate = datetime.date(year = 2030, month = 12, day = 30)#sets maximum date on the calendar
```

Fig. 12 dateEntryPage(*args) and getDateValues()

```
#commands to retrive an image from the system which will be used to design the various widgets present in the GUI
selDate_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/OtherDates/selectdate.png")
Enter_Button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/OtherDates/Enter_Button.png")
set_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/OtherDates/Set_Button.png")
selDate_Background = ImageTk.PhotoImage(selDate_back_img)
Enter_Button = ImageTk.PhotoImage(Enter_Button_img)
Set_Button = ImageTk.PhotoImage(set_button_img)

s = ttk.Style()
s.theme_use('clam')

#initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at an arbitrary location on the window
customFrame = ttk.Frame(root, padding = "0")
background_label = Label(root, image=selDate_Background)
background_label.image = selDate_Background # this is to keep a copy of the image in the file
enterButton = Button(root, text = "", command = curStoresPage, image = Enter_Button)
setButton = Button(root, text = "Set", command = getDateValues, image = Set_Button)
backButton = Button(root, text = "Go Back", command = mainScreen, image = back_button)
cal = Calendar(root, font="Arial 14", selectmode='day', locale='en_US', mindate=mindate, maxdate=maxdate, disabledforeground='red', cursor="hand1", year=2018, mont
hourEntry = ttk.Combobox(root, values = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23), width = 10, textvariable = hourStri
hourEntry.current(9)#sets the default value of the combobox
minuteEntry = ttk.Combobox(root, values = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 3
minuteEntry.current(29)#sets default value of combobox
endProg = Button(root, text = "", command = leavePage, image = CloseButton)
#Placement block for the widgets to be displayed to the user. Defines the coordinates on the window at which each widget should be placed
customFrame.place()
background_label.place(x=0, y=0, relwidth=1, relheight=1)
enterButton.place(x=650, y=340, height = 146, width = 128)
setButton.place(x=520, y=340, height=146, width=128)
backButton.place(x=30, y=400, height=50, width=50)
cal.pack(expand=False)
cal.place(x = 250, y = 150)
hourEntry.place(x=250, y=380)
minuteEntry.place(x=420, y=380)
endProg.place(x=740, y=20, height =34, width=40)
```

Fig. 13 Visual Enhancements

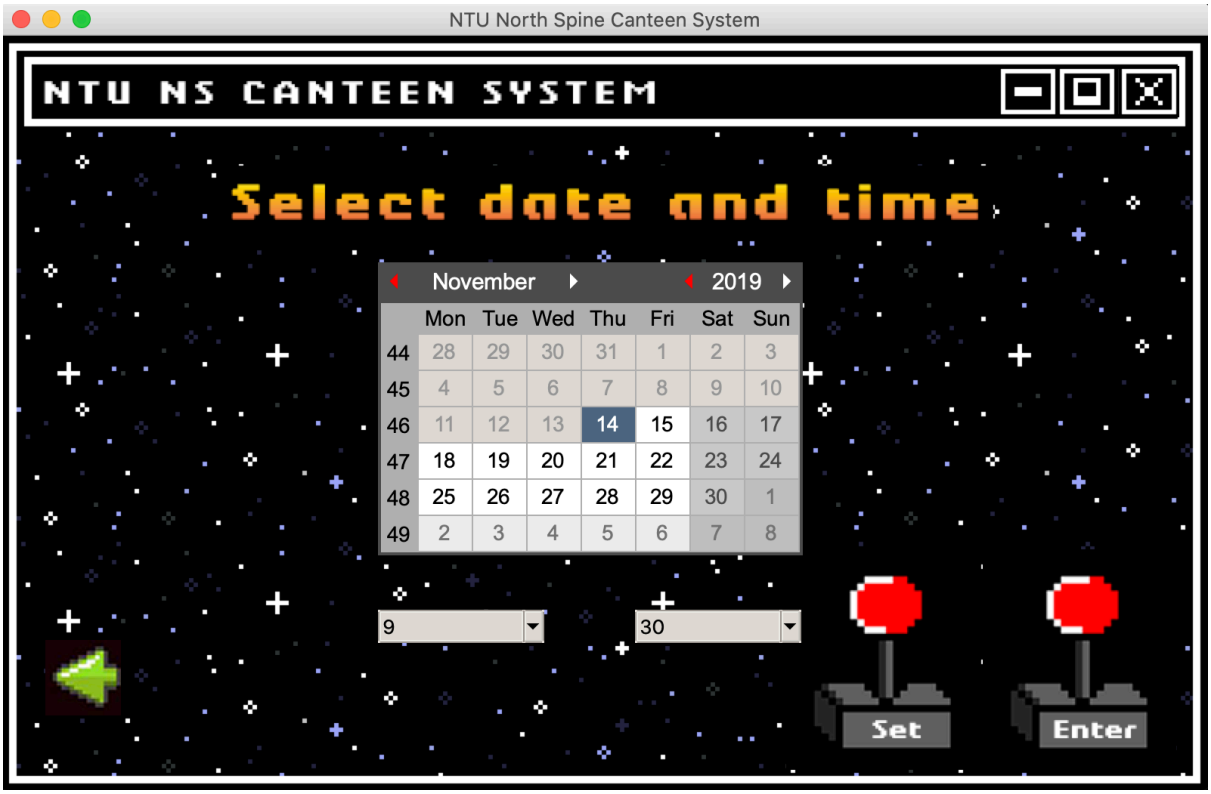


Fig. 14 GUI of dateEntryPage(*args)

5.2.7 popup_showinfo()

This function allows the program to display a popup window (Fig. 15).

```
#Function to display a popup window
def popup_showinfo():
    showinfo("Error", "Plese enter a valid input!")
```

Fig. 15 popup_showinfo()

5.2.8 curStoresPage() and mainScreen(*args)

curStoresPage() and mainScreen is the backbone of the program which allows users to select either the current date or view the available stores at the current time (Fig. 16 & 17). Images are retrieved from the system to enhance the visuals, and stylings are done intricately to ensure that each and every label and buttons are placed appropriately (Fig. 18 & 19)

```
def curStoresPage(*args):
    global flag
    if flag == False:
        SetDateToCurr()
    else:
        pass

    #commands to retrive an image from the system which will be used to design the various widgets present in the GUI
    availableStores_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/AvailStores.png")
    kfc_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/KFC.png")
    McD_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/McD.png")
    noodles_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/Noodles.png")
    western_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/Western.png")
    miniwok_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/Miniwok.png")
    AvailStores = ImageTk.PhotoImage(availableStores_back_img)
    KFC = ImageTk.PhotoImage(kfc_button_img)
    McD = ImageTk.PhotoImage(McD_button_img)
    Noodles = ImageTk.PhotoImage(noodles_button_img)
    PorkChop = ImageTk.PhotoImage(western_button_img)
    Miniwok = ImageTk.PhotoImage(miniwok_button_img)

    style = ttk.Style()#style tag to define the label style and font color
    style.configure("TLabel", foreground = 'black')
```

```
background_label = Label(root, image=AvailStores)
background_label.image = AvailStores # this is to keep a copy of the image in the file
store1 = Button(root, text = "KFC", command = kfcRegular, image = KFC)
store2 = Button(root, text = "McDonald's", command = McDRegular, image = McD)
store3 = Button(root, text = "Mini Wok", command = Chinese, image = Miniwok)
store4 = Button(root, text = "Noodle", command = Noodle, image = Noodles)
store5 = Button(root, text = "Western", command = Western, image = PorkChop)
backButton = Button(root, text = "", command = mainScreen, image = back_button)
endProg = Button(root, text = "", command = leavePage, image = CloseButton)

#placement block for the widgets to be displayed to the user. Defines the coordinates on the window at which each widget should be placed
background_label.place(x=0, y=0, relwidth=1, relheight=1)
store1.place(x=350,y=175, height=46, width=190)
store2.place(x=350,y=230, height=46, width=190)
store3.place(x=350, y=405, height=46, width=190)
store4.place(x=350, y=290, height=46, width=190)
store5.place(x=350, y=348, height=46, width=190)
backButton.place(x=30, y=400, height=50, width=50)
endProg.place(x=740, y=20, height =34, width=40)
```

Fig. 16 curStoresPage(*args)

```
def mainScreen(*args):
    global flag
    flag = False

    style = ttk.Style()#style tag to define the label style and font color
    style.configure("TLabel", foreground = 'white')

    #commands to retrive an image from the system which will be used to design the various widgets present in the GUI
    main_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/MainBackground.png")
    currStores_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/CurrentStore.png")
    otherDates_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/OtherDates.png")
    image3 = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/black.png")
    MainBackground = ImageTk.PhotoImage(main_back_img)
    CurrentStore = ImageTk.PhotoImage(currStores_button_img)
    OtherDates = ImageTk.PhotoImage(otherDates_button_img)
    black_background = ImageTk.PhotoImage(image3)

    current_time = time_str
    day = day_str
```

```
#initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at an arbitrary location on the window
background_label = Label(root, image=MainBackground)
background_label.image = MainBackground # this is to keep a copy of the image in the file
dayLabel = ttk.Label(root, text = day, image = black_background, compound=CENTER)
timeLabel = ttk.Label(root, text = current_time, image = black_background, compound=CENTER)
currDate = Button(root, text = "Current Stores", command = curStoresPage, image = CurrentStore)
otherDate = Button(root, text = "View stores by other dates", command = dateEntryPage, image = OtherDates)
endProg = Button(root, text = "", command = leavePage, image = CloseButton)

#placement block for the widgets to be displayed to the user. Defines the coordinates on the window at which each widget should be placed
background_label.place(x=0, y=0, relwidth=1, relheight=1)
dayLabel.place(x=100, y=20, height = 30, width = 100)
timeLabel.place(x=400, y=20, height = 30, width = 100)
currDate.place(x=252, y=270, height=54, width=292)
otherDate.place(x=252, y=350, height=54, width=292)
endProg.place(x=740, y=20, height =34, width=40)

root.mainloop()
```

Fig. 17 mainScreen(*args)

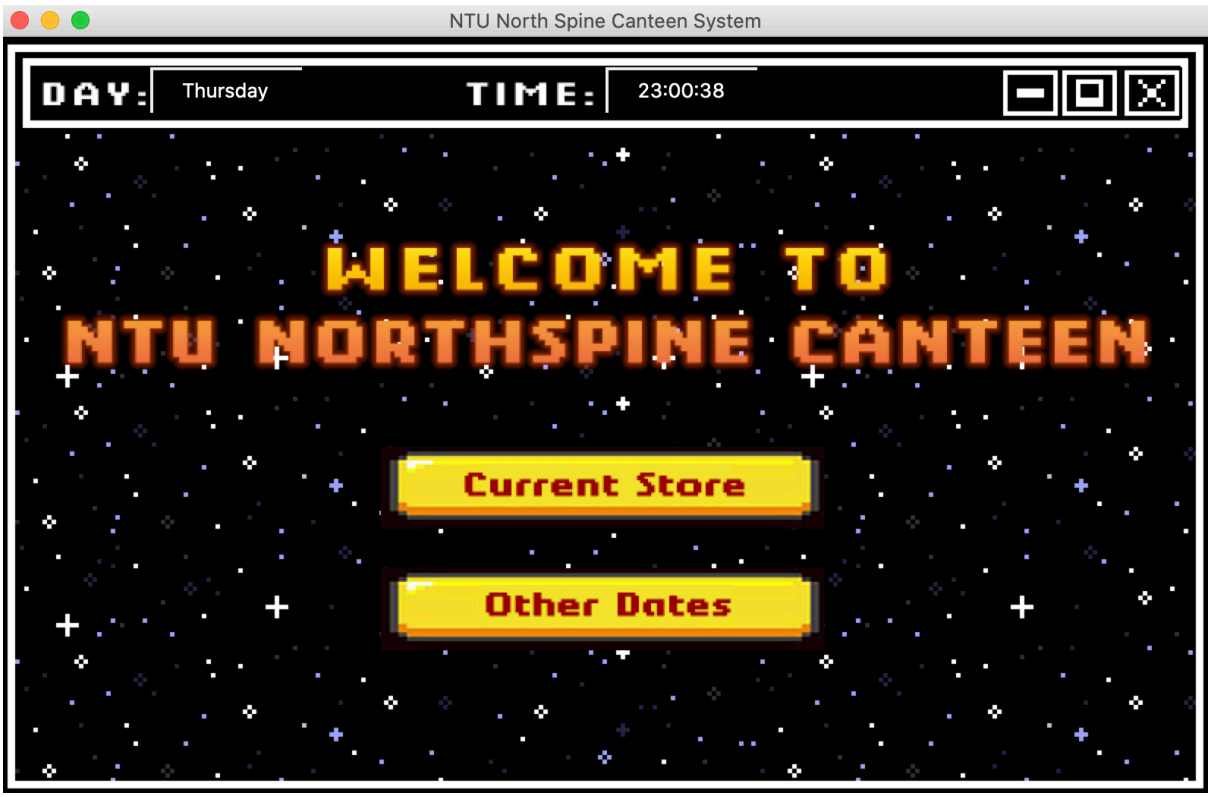


Fig. 18 GUI for mainScreen(*args)

Fig. 19 GUI for curStoresPage(*args)

5.2.9 calcWaitTime(*args)

This function is for calculating the waiting time and is defined by the algorithm using the number of people in queue (Fig. 20). If the user input the value, the waiting time will be calculated as multiplying the value by 1.5. The system only works when the value is in the right data type, which is 'int'. Otherwise, the pop-up message will appear. User can re-enter the value until an integer of value more than 0 is read.

```
def calcWaitTime(*args):#function to retrieve the number of people in queue and set the total wait time depending on user input
    try:
        value = int(numPeople.get())
        if value < 0:
            popup_showinfo()
        else:
            waitTime.set(value*1.5)
    except ValueError:
        popup_showinfo()
```

Fig. 20 calcWaitTime(*args)

5.2.10 Store Function

This function displays each store's menu based on the time and provides the system for calculating the waiting time. Each function is defined by the name of the stores. (i.e Noodle, Western, Chinese, KFC, McDonald's) (Fig. 21).

```
#noodle store GUI
def Noodle(*args):
    itemDict = {}#initialization of dictionary to hold store menu information

    #commands to retrive an image from the system which will be used to design the various widgets present in the GUI
    noodle_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/Noodles_main.png")
    closed_screen_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/Noodles_close.png")
    Noodles_main = ImageTk.PhotoImage(noodle_back_img)
    Noodles_close = ImageTk.PhotoImage(closed_screen_img)

    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at an arbitrary location on the window
    background_label = Label(root, image=Noodles_main)
    background_label.image = Noodles_main # this is to keep a copy of the image in the file
    calcButton = Button(root, text = "Calculate", command = calcWaitTime, image=EnterButton)
    waittime_result = ttk.Label(root, textvariable = waitTime)
    qEntry = ttk.Entry(root, textvariable = numPeople)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg)
#the conditional statements retrieve store menu information from text files depending on the present or user defined date and time
if hour_int >= 8 and hour_int < 20:
    if(day_str == "Monday" or day_str == "Wednesday" or day_str == "Friday"):
        f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/noodle_mwf.txt", "r")
    else:
        f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/noodle_tt.txt", "r")
else:
    noodles_close_label = ttk.Label(root, image= Noodles_close)
    noodles_close_label.place(x=0, y=0, relwidth=1, relheight=1)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    backButton.place(x=30, y=400, height=50, width=50)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)
    endProg.place(x=740, y=20, height =34, width=40)

display_menu(f, itemDict)

root.bind('<Return>', calcWaitTime)#binds the calculate wait time button to the enter key on the keyboard
```

Fig. 21 Noodles(*args)

The menu information are stored in the dictionary, 'itemDict{}'. As some stores provide different menu for each day and/or time, the different text files of the menu will be opened accordingly based on the time or day the user defined. For example, different menu of Noodle will be displayed on Monday and Tuesday. Comparing the user-defined time with the opening hours, the user gets menu information or the

image of ‘closed’. If the user gets the ‘closed’ image, the back button is allowed to return to the page of the current stores.

6. Program Testing

Using exception handling, we are able to handle bad inputs from users (Fig. 22).

```
def calcWaitTime(*args): #function to retrieve the number of people in queue and set the total wait time depending on user input
    try:
        value = int(numPeople.get())
        waitTime.set(value*1.5)
    except ValueError:
        popup_showinfo()
```

Fig. 22 Exception Handling

When user enters invalid inputs such as no input, strings, floats, or negative integer, a popup window will prompt the user to enter the correct input (Fig. 23 & 24).

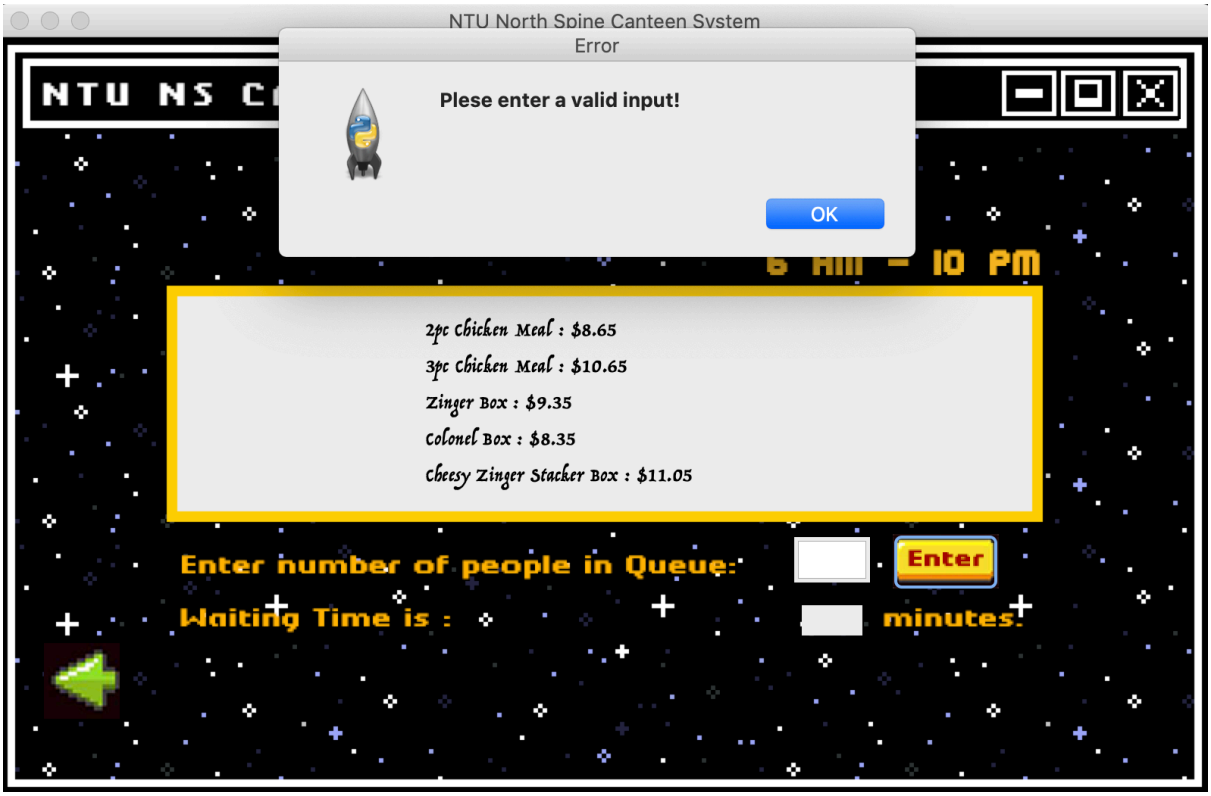


Fig. 23 User enters no input

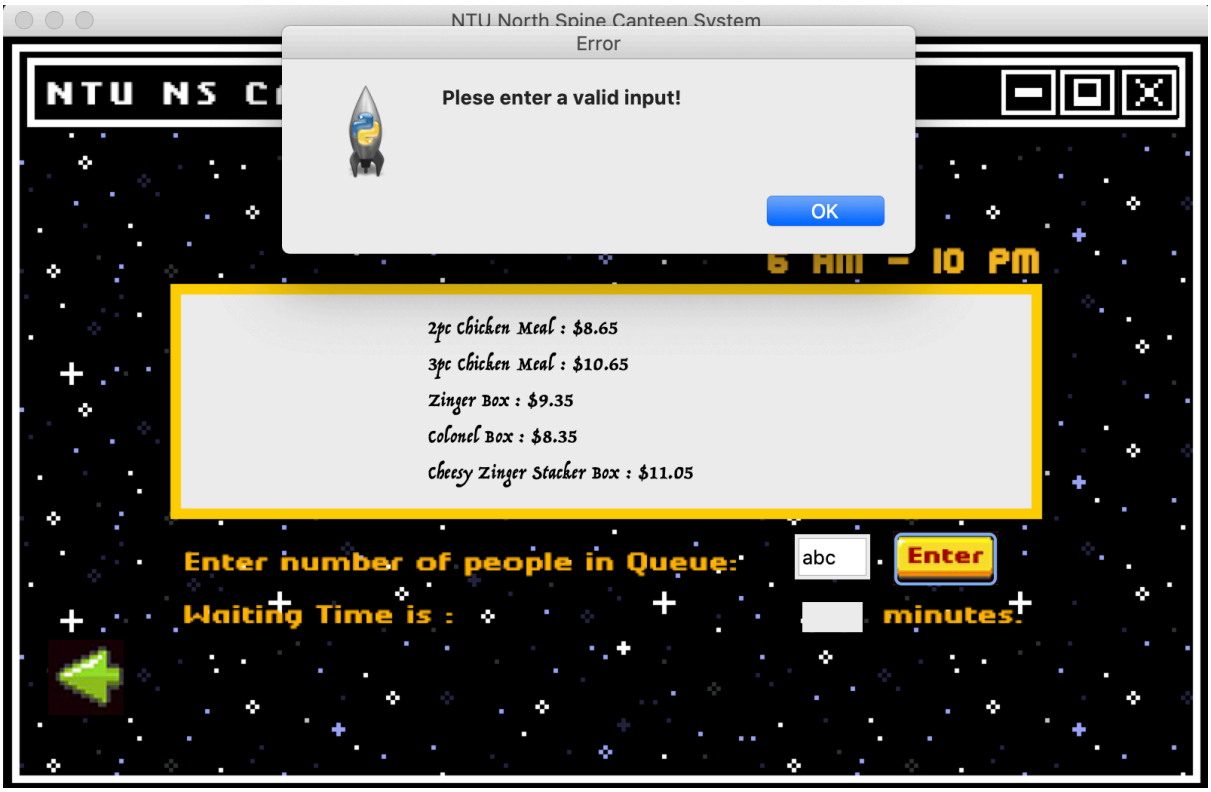


Fig. 24 User enters a string

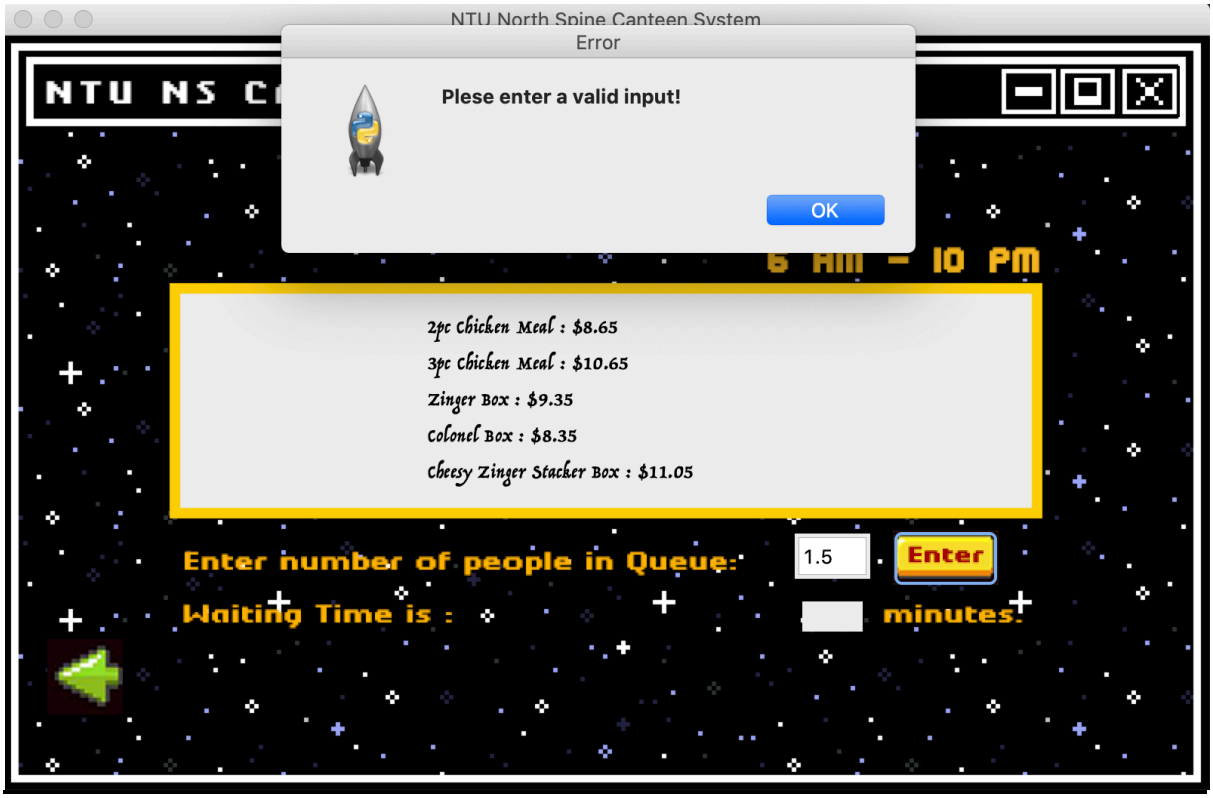


Fig. 25 User enters a float

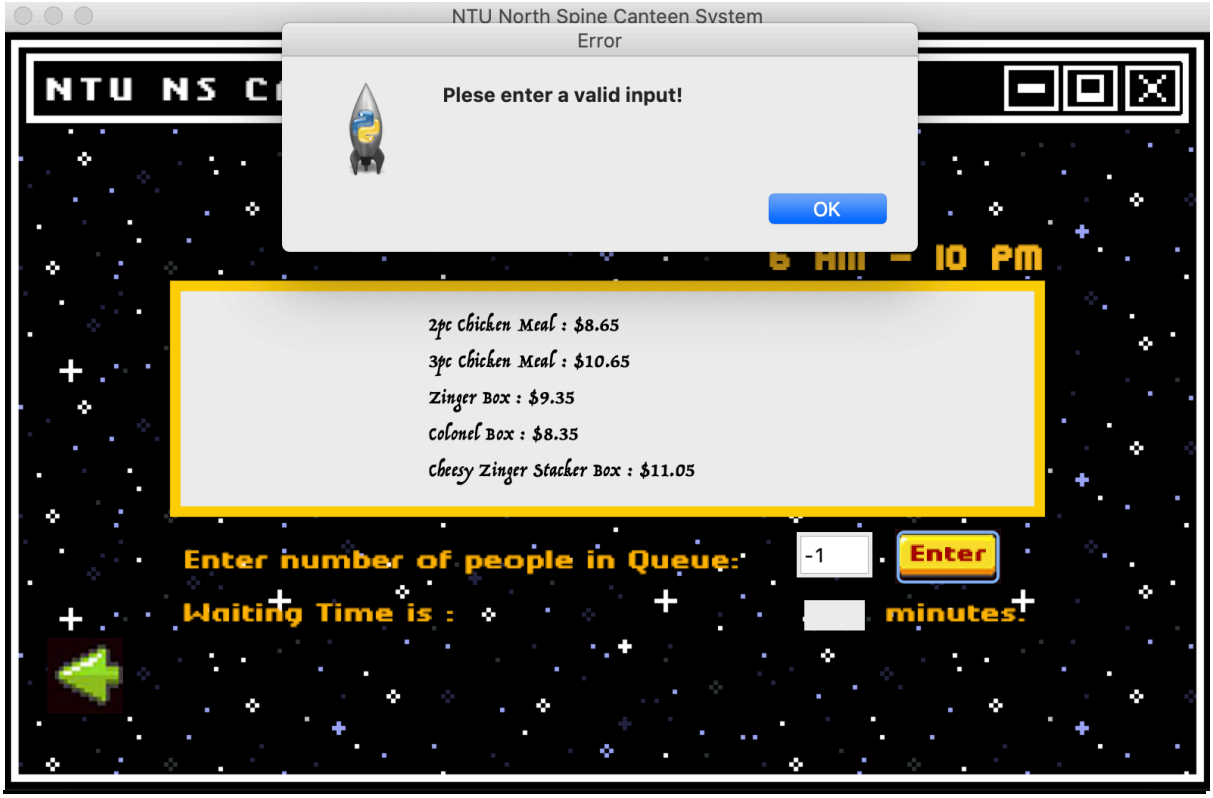


Fig.26 User enters negative integer

7. Source Code

```
#Program to view stores, menu, store timings for various eateries in the north spine canteen

from tkinter import *
from tkinter import ttk
from PIL import ImageTk, Image
from tkmacosx import Button
try:
    import tkinter as tk
    from tkinter import ttk
except ImportError:
    import Tkinter as tk
    import ttk

from tkcalendar import Calendar, DateEntry
from tkinter.messagebox import showinfo
#the above import statements deal with creation and design of GUI using tkinter

import sys
import datetime #python library to retrieve and manipulate system and user defined date and time

def popup_showinfo():
    showinfo("Error", "Plese enter a valid input!")

def calcWaitTime(*args): #function to retrieve the number of people in queue and set the total wait time
depending on user input
```

```
    try:
        value = float(numPeople.get())
        waitTime.set(value*1.5)
    except ValueError:
        popup_showinfo()

#function to define the placement of widgets at each storefront page
def place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg):
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    calcButton.place(x=590, y=328, height=36, width=70)
    waittime_result.place(x=530,y=375, height=20, width=40)
    qEntry.place(x=525,y=330, height=30, width=50)
    backButton.place(x=30, y=400, height=50, width=50)
    endProg.place(x=740, y=20, height =34, width=40)

#function to display the menu items of a particular store
def display_menu(f, itemDict):
    line_str = f.readlines() #initialization of string which holds text file contents

    for i in line_str: #converts the contents of the string into a dictionary with the menu item as the key and the
price as the value
        new_str = i.split(":")
        itemDict[new_str[0]] = new_str[1]

    final_str = "" #initialization of final string that will be used to display the menu label widget
    for key, value in itemDict.items():
        final_str += key + " : " + value

    itemLabel = ttk.Label(root, text = final_str, font=('Trattatello')) #menu items label initialization and styling
    itemLabel.place(x=280,y=180, width=250, height=125)

#Function to set the program to the current system date and time
def SetDateToCurr():
    global day_str
    global time_str
    global hour_int
    now = datetime.datetime.now()
    day_str = datetime.datetime.today().strftime("%A")
    time_str = now.strftime("%H:%M:%S")
    hour_int = int(now.hour)

#Function to set the program to a user defined date and time
def SetDateToCustom(cus_day, cus_hour, cus_min):
    global day_str
    global time_str
    global hour_int

    #Special case handling to display the user defined time properly when it is a single digit
    if int(cus_hour) < 10:
        cus_hour = "0" + cus_hour
```

```
    if int(cus_min) < 10:
        cus_min = "0" + cus_min

    day_str = cus_day
    hour_int = cus_hour
    time_str = cus_hour + ":" + cus_min + ":00"
    hour_int = int(cus_hour)

#Function to end program when user clicks exit
def leavePage():
    try:
        root.destroy()
        exit()
    except:
        pass

#noodle store GUI
def Noodle(*args):
    itemDict = {}#initialization of dictionary to hold store menu information

    #commands to retrive an image from the system which will be used to design the various widgets present in
the GUI
    noodle_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/Stores/Noodles_main.png")
    closed_screen_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/Stores/Noodles_close.png")
    Noodles_main = ImageTk.PhotoImage(noodle_back_img)
    Noodles_close = ImageTk.PhotoImage(closed_screen_img)

    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at
an arbitrary location on the window
    background_label = Label(root, image=Noodles_main)
    background_label.image = Noodles_main # this is to keep a copy of the image in the file
    calcButton = Button(root, text = "Calculate", command = calcWaitTime, image=EnterButton)
    waittime_result = ttk.Label(root, textvariable = waitTime)
    qEntry = ttk.Entry(root, textvariable = numPeople)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg)

    #the conditional statements retrieve store menu information from text files depending on the present or user
defined date and time
    if hour_int >= 8 and hour_int < 20:
        if(day_str == "Monday" or day_str == "Wednesday" or day_str == "Friday"):
            f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/noodle_mwf.txt", "r")
        else:
            f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/noodle_tt.txt", "r")
    else:
        noodles_close_label = ttk.Label(root, image= Noodles_close)
        noodles_close_label.place(x=0, y=0, relwidth=1, relheight=1)
```

```
        backButton = Button(root, text = "", command = curStoresPage, image = back_button)
        backButton.place(x=30, y=400, height=50, width=50)
        endProg = Button(root, text = "", command = leavePage, image = CloseButton)
        endProg.place(x=740, y=20, height =34, width=40)

    display_menu(f, itemDict)

    root.bind('<Return>', calcWaitTime)#binds the calculate wait time button to the enter key on the keyboard

#western store GUI
def Western(*args):
    itemDict = {}#initialization of dictionary to hold store menu information

    #commands to retrive an image from the system which will be used to design the various widgets present in
the GUI
    western_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/Stores/Western_main.png")
    store_closed_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/Stores/Western_close.png")
    Western_main = ImageTk.PhotoImage(western_back_img)
    Western_close = ImageTk.PhotoImage(store_closed_img)

    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at
an arbitrary location on the window
    background_label = Label(root, image=Western_main)
    background_label.image = Western_main # this is to keep a copy of the image in the file
    calcButton = Button(root, text = "Calculate", command = calcWaitTime, image=EnterButton)
    waittime_result = ttk.Label(root, textvariable = waitTime)
    qEntry = ttk.Entry(root, textvariable = numPeople)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg)

    #the conditional statements retrieve store menu information from text files depending on the present or user
defined date and time
    if hour_int >= 8 and hour_int < 20:
        if(day_str == "Monday" or day_str == "Wednesday" or day_str == "Friday"):
            f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/western_mwf.txt", "r")
        else:
            f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/western_tt.txt", "r")
    else:
        Western_close_label = ttk.Label(root, image= Western_close)
        Western_close_label.place(x=0, y=0, relwidth=1, relheight=1)
        backButton = Button(root, text = "", command = curStoresPage, image = back_button)
        backButton.place(x=30, y=400, height=50, width=50)
        endProg = Button(root, text = "", command = leavePage, image = CloseButton)
        endProg.place(x=740, y=20, height =34, width=40)

    display_menu(f, itemDict)
```

```
    root.bind('<Return>', calcWaitTime)#binds the caluclate wait time button to the enter key on the keyboard

#Chinese store GUI
def Chinese(*args):
    itemDict = {}#initialization of dictionary to hold store menu information

    #commands to retrive an image from the system which will be used to design the various widgets present in the GUI
    miniwok_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/Miniwok_Main.png")
    store_closed_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/Miniwok_close.png")
    Miniwok_Main = ImageTk.PhotoImage(miniwok_back_img)
    Miniwok_close = ImageTk.PhotoImage(store_closed_img)

    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at an arbitrary location on the window
    background_label = Label(root, image=Miniwok_Main)
    background_label.image = Miniwok_Main # this is to keep a copy of the image in the file
    calcButton = Button(root, text = "Calculate", command = calcWaitTime, image=EnterButton)
    waittime_result = ttk.Label(root, textvariable = waitTime)
    qEntry = ttk.Entry(root, textvariable = numPeople)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg)

    #the conditional statements retrieve store menu information from text files depending on the present or user defined date and time
    if hour_int >= 8 and hour_int < 20:
        if(day_str == "Monday" or day_str == "Wednesday" or day_str == "Friday"):
            f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/chinese_mwf.txt", "r")
        else:
            f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/chinese_tt.txt", "r")
    else:
        Miniwok_close_label = ttk.Label(root, image= Miniwok_close)
        Miniwok_close_label.place(x=0, y=0, relwidth=1, relheight=1)
        backButton = Button(root, text = "", command = curStoresPage, image = back_button)
        backButton.place(x=30, y=400, height=50, width=50)
        endProg = Button(root, text = "", command = leavePage, image = CloseButton)
        endProg.place(x=740, y=20, height =34, width=40)

    display_menu(f, itemDict)

    root.bind('<Return>', calcWaitTime)#binds the caluclate wait time button to the enter key on the keyboard

#McD store GUI
def McDRegular(*args):
    itemDict = {}#initialization of dictionary to hold store menu information
```

```
    #commands to retrive images from the system which will be used to design the various widgets present in the GUI
    mcd_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/MCD_main.png")
    store_closed_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/MCD_close.png")
    MCD_main = ImageTk.PhotoImage(mcd_back_img)
    MCD_close = ImageTk.PhotoImage(store_closed_img)


    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at an arbitrary location on the window
    background_label = Label(root, image=MCD_main)
    background_label.image = MCD_main # this is to keep a copy of the image in the file
    calcButton = Button(root, text = "Calculate", command = calcWaitTime, image=EnterButton)
    waittime_result = ttk.Label(root, textvariable = waitTime)
    qEntry = ttk.Entry(root, textvariable = numPeople)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg)


    #the conditional statements retrieve store menu information from text files depending on the present or user defined date and time
    if(hour_int >= 7 and hour_int <= 10):
        f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/mcd_breakfast.txt", "r")
    elif hour_int >= 11 and hour_int < 22:
        f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/mcd_regular.txt", "r")
    else:
        MCD_close_label = ttk.Label(root, image= MCD_close)
        MCD_close_label.place(x=0, y=0, relwidth=1, relheight=1)
        backButton = Button(root, text = "", command = curStoresPage, image = back_button)
        backButton.place(x=30, y=400, height=50, width=50)
        endProg = Button(root, text = "", command = leavePage, image = CloseButton)
        endProg.place(x=740, y=20, height =34, width=40)

    display_menu(f, itemDict)

    root.bind('<Return>', calcWaitTime)#binds the caluclate wait time button to the enter key on the keyboard

#kfc store GUI
def kfcRegular(*args):
    itemDict = {}#initialization of dictionary to hold store menu information
    global time_str
    global day_str

    #commands to retrive an image from the system which will be used to design the various widgets present in the GUI
    kfc_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/KFC_main.png")
```

```
    store_closed_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/Stores/KFC_close.png")
    KFC_main = ImageTk.PhotoImage(kfc_back_img)
    KFC_close = ImageTk.PhotoImage(store_closed_img)


    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at
an arbitrary location on the window
    background_label = Label(root, image=KFC_main)
    background_label.image = KFC_main # this is to keep a copy of the image in the file
    calcButton = Button(root, text = "Calculate", command = calcWaitTime, image=EnterButton)
    waittime_result = ttk.Label(root, textvariable = waitTime)
    qEntry = ttk.Entry(root, textvariable = numPeople)
    backButton = Button(root, text = "", command = curStoresPage, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    place_widgets(background_label, calcButton, waittime_result, qEntry, backButton, endProg)

    #the conditional statements retrieve store menu information from text files depending on the present or user
defined date and time
    if(hour_int >= 6 and hour_int <= 10):
        f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/kfc_breakfast.txt", "r")
    elif hour_int >= 11 and hour_int < 22:
        f = open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/kfc_regular.txt", "r")
    else:
        KFC_close_label = ttk.Label(root, image= KFC_close)
        KFC_close_label.place(x=0, y=0, relwidth=1, relheight=1)
        backButton = Button(root, text = "", command = curStoresPage, image = back_button)
        backButton.place(x=30, y=400, height=50, width=50)
        endProg = Button(root, text = "", command = leavePage, image = CloseButton)
        endProg.place(x=740, y=20, height =34, width=40)

    display_menu(f, itemDict)

    root.bind('<Return>', calcWaitTime)#binds the caluclate wait time button to the enter key on the keyboard

#Defines a page that allows users to input custom date and time preferences
def dateEntryPage(*args):
    hourString = StringVar()
    minString = StringVar()

    #Function to retrieve the custom date and time values from the user
    def getDateValues():
        try:
            day = cal.selection_get()
            hour = hourString.get()
            minute = minString.get()
            SetDateToCustom(day.strftime("%A"), hour, minute)
        except ValueError:
            pass
```

```
    global flag
    flag = True

    today = datetime.date.today()
    mindate = datetime.date(today.year, today.month, today.day)#sets minimum date on the calendar as todays
date
    maxdate = datetime.date(year = 2030, month = 12, day = 30)#sets maximum date on the calendar

    #commands to retrive an image from the system which will be used to design the various widgets present in
the GUI
    selDate_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/OtherDates/selectdate.png")
    Enter_Button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/OtherDates/Enter_Button.png")
    set_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini
Project/NSCanteenSystem/OtherDates/Set_Button.png")
    selDate_Background = ImageTk.PhotoImage(selDate_back_img)
    Enter_Button = ImageTk.PhotoImage(Enter_Button_img)
    Set_Button = ImageTk.PhotoImage(set_button_img)

    s = ttk.Style()
    s.theme_use('clam')

    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at
an arbitrary location on the window
    customFrame = ttk.Frame(root, padding = "0")
    background_label = Label(root, image=selDate_Background)
    background_label.image = selDate_Background # this is to keep a copy of the image in the file
    enterButton = Button(root, text = "", command = curStoresPage, image = Enter_Button)
    setButton = Button(root, text = "Set", command = getDateValues, image = Set_Button)
    backButton = Button(root, text = "Go Back", command = mainScreen, image = back_button)
    cal = Calendar(root, font="Arial 14", selectmode='day', locale='en_US', mindate=mindate, maxdate=maxdate,
disabledforeground='red', cursor="hand1", year=2018, month=2, day=5)
    hourEntry = ttk.Combobox(root, values = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23), width = 10, textvariable = hourString, state="readonly")
    hourEntry.current(9)#sets the default value of the combobox
    minuteEntry = ttk.Combobox(root, values = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
51, 52, 53, 54, 55, 56, 57, 58, 59), width = 10, textvariable = minString, state="readonly")
    minuteEntry.current(29)#sets default value of combobox
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    #Placement block for the widgets to be displayed to the user. Defines the coordinates on the window at which
each widget should be placed
    customFrame.place()
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    enterButton.place(x=650, y=340, height = 146, width = 128)
    setButton.place(x=520, y=340, height=146, width=128)
    backButton.place(x=30, y=400, height=50, width=50)
    cal.pack(expand=False)
    cal.place(x = 250, y = 150)
```

```
    hourEntry.place(x=250, y=380)
    minuteEntry.place(x=420, y=380)
    endProg.place(x=740, y=20, height =34, width=40)

def curStoresPage(*args):
    global flag
    if flag == False:
        SetDateToCurr()
    else:
        pass

    #commands to retrieve an image from the system which will be used to design the various widgets present in the GUI
    availableStores_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/AvailStores.png")
    kfc_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/KFC.png")
    McD_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/McD.png")
    noodles_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/Noodles.png")
    western_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/Western.png")
    miniwok_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/CurrStores/Miniwok.png")
    AvailStores = ImageTk.PhotoImage(availableStores_back_img)
    KFC = ImageTk.PhotoImage(kfc_button_img)
    McD = ImageTk.PhotoImage(McD_button_img)
    Noodles = ImageTk.PhotoImage(noodles_button_img)
    PorkChop = ImageTk.PhotoImage(western_button_img)
    Miniwok = ImageTk.PhotoImage(miniwok_button_img)

    style = ttk.Style()#style tag to define the label style and font color
    style.configure("TLabel", foreground = 'black')

    background_label = Label(root, image=AvailStores)
    background_label.image = AvailStores # this is to keep a copy of the image in the file
    store1 = Button(root, text = "KFC", command = kfcRegular, image = KFC)
    store2 = Button(root, text = "McDonald's", command = McDRegular, image = McD)
    store3 = Button(root, text = "Mini Wok", command = Chinese, image = Miniwok)
    store4 = Button(root, text = "Noodle", command = Noodle, image = Noodles)
    store5 = Button(root, text = "Western", command = Western, image = PorkChop)
    backButton = Button(root, text = "", command = mainScreen, image = back_button)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    #placement block for the widgets to be displayed to the user. Defines the coordinates on the window at which each widget should be placed
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    store1.place(x=350,y=175, height=46, width=190)
    store2.place(x=350,y=230, height=46, width=190)
    store3.place(x=350, y=405, height=46, width=190)
```

```
    store4.place(x=350, y=290, height=46, width=190)
    store5.place(x=350, y=348, height=46, width=190)
    backButton.place(x=30, y=400, height=50, width=50)
    endProg.place(x=740, y=20, height =34, width=40)

def mainScreen(*args):
    global flag
    flag = False

    style = ttk.Style()#style tag to define the label style and font color
    style.configure("TLabel", foreground = 'white')

    #commands to retrive an image from the system which will be used to design the various widgets present in the GUI
    main_back_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/MainBackground.png")
    currStores_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/CurrentStore.png")
    otherDates_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/OtherDates.png")
    image3 = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/MainPage/black.png")
    MainBackground = ImageTk.PhotoImage(main_back_img)
    CurrentStore = ImageTk.PhotoImage(currStores_button_img)
    OtherDates = ImageTk.PhotoImage(otherDates_button_img)
    black_background = ImageTk.PhotoImage(image3)

    current_time = time_str
    day = day_str

    #initialization block for the widgets to be displayes to the user. Defines the various widgets and places them at an arbitrary location on the window
    background_label = Label(root, image=MainBackground)
    background_label.image = MainBackground # this is to keep a copy of the image in the file
    dayLabel = ttk.Label(root, text = day, image = black_background, compound=CENTER)
    timeLabel = ttk.Label(root, text = current_time, image = black_background, compound=CENTER)
    currDate = Button(root, text = "Current Stores", command = curStoresPage, image = CurrentStore)
    otherDate = Button(root, text = "View stores by other dates", command = dateEntryPage, image = OtherDates)
    endProg = Button(root, text = "", command = leavePage, image = CloseButton)

    #placement block for the widgets to be displayed to the user. Defines the coordinates on the window at which each widget should be placed
    background_label.place(x=0, y=0, relwidth=1, relheight=1)
    dayLabel.place(x=100, y=20, height = 30, width = 100)
    timeLabel.place(x=400, y=20, height = 30, width = 100)
    currDate.place(x=252, y=270, height=54, width=292)
    otherDate.place(x=252, y=350, height=54, width=292)
    endProg.place(x=740, y=20, height =34, width=40)

    root.mainloop()
```

```
flag = False #flag variable to check if the store should display the current date and time or the custom date and time
hour_int = "" #String to store current or user defined hour
day_str = "" #String to store current or user defined day
time_str = "" #String to store current or user defined time

now = datetime.datetime.now()
day_str = datetime.datetime.today().strftime("%A")#Returns the current day as a string
time_str = now.strftime("%H:%M:%S")#Returns the current time as HH:MM:SS
hour_int = int(now.hour)

root = Tk()#intial root window on which information and widgets will be displayed
root.title("NTU North Spine Canteen System")
root.geometry("800x504+325+175")#sets the position of the window on execution of the program
root.maxsize(800,500)#sets the maximum size of the GUI window to prevent resizing
root.minsize(800,500)#sets the minimum size of the GUI window to prevent resizing

waitTime = StringVar()#initializes a string that will store the waiting time at the store
numPeople = StringVar()#initializes a string that will retrieve the number of people in queue at the store

#public declaration and retrieval of images that will be used in every page of the GUI
back_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/back_button.png")
close_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/CloseButton.png")
enter_button_img = Image.open("/Users/jonathanchan/Desktop/CZ1003/Mini Project/NSCanteenSystem/Stores/Enter.png")
back_button = ImageTk.PhotoImage(back_button_img)
CloseButton = ImageTk.PhotoImage(close_button_img)
EnterButton = ImageTk.PhotoImage(enter_button_img)

mainScreen()
```

8. Reflections

The development of North Spine Canteen System has allowed our group to go in depth with not only Python, but also with the Tkinter module.

With dynamism in mind, we have designed the program in such a way that it is futureproof; whereby any changes in the stores' menu can be easily changed without modifying a huge part of the program. Factors which are variable include store names, menu, and the price. With this, the overall reusability of this whole program is much more effective, and this also allows for easier debugging as most parts of the code will remain unchanged should there be a need for any changes to the program's content.

Difficulties faced during this course:

- The course provided the base knowledge for a wide range of concepts required for computational thinking. A lot of these concepts required us to go beyond the scope of this course for its effective implementation and design.

Learning Outcomes:

- Although we had to seek various other resources to complement our learning in this course, we believe this was very beneficial since it taught us how to take an abstract concept and use various resources available to us, such as the internet, to find its various implementations and focus on the most effective ones.
- Being part of a team for the mini project at the end of this course gave us the chance to work with individuals that have varied programming methods and styles. This allowed us to witness how the integration of our individual ideas and implementations can be used to deliver a more polished end product.

Further improvement suggestions:

- The course could include more practice tasks to reinforce the concepts that we learn in the online lectures.
- While working on the mini project, we realized that there were several concepts of object oriented programming that we have not been introduced to. Since python is an OOP language, an introduction to such concepts would have played in our favour during the course of the assignment.

9. Job Allocation

Aneez	Heon Jung Kim	Jonathan
-Implemented the design to ensure that it works with the flow of the program -Worked on dateEntryPage, SetDatetoCur and SetDateToCustom -Worked on the individual stores pages	-Designing the visuals of the program; including the creation of backgrounds, buttons etc. -Worked on CurStoresPage, calcWaitTime, display_menu and leavePage	-Implemented the design to ensure that it works with the flow of the program -Worked on mainScreen, popup_showinfo and place_widgets -Worked on the individual stores pages

10. Bibliography:

- https://pypi.org/project/tkcalendar/ - An Introduction to calendars in Tkinter and its usage
- https://tkdocs.com/ - Tkinter Documentation for python
- https://stackoverflow.com/ - An online forum to present and solve developer related problems